Assignment Name : Regression and Its Evaluation

Question 1: What is Simple Linear Regression?

Answer:

Simple Linear Regression is a statistical method that models the relationship between two variables: one independent variable (X) and one dependent variable (Y). It assumes that the relationship between X and Y is linear and can be represented by the equation:

$$Y=\beta 0+\beta 1X+\epsilon$$

where:

- $\beta 0$= intercept
- $\beta 1$= slope of the regression line
- ϵ= error term

It is mainly used for prediction and understanding the strength of the relationship between variables.

Question 2: What are the key assumptions of Simple Linear Regression?

Answer:

1. **Linearity** – The relationship between X and Y is linear.
2. **Independence** – Observations are independent of each other.
3. **Homoscedasticity** – The variance of residuals is constant across all levels of X.
4. **Normality of Errors** – The residuals are normally distributed.
5. **No multicollinearity** – Since it involves only one predictor, multicollinearity is not applicable.

Question 3: What is heteroscedasticity, and why is it important to address in regression models?

Answer:

Heteroscedasticity occurs when the variance of residuals is not constant across all levels of the independent variable(s).

- **Why it is important to address?**
    - It violates the assumption of homoscedasticity.

- Leads to inefficient estimates of regression coefficients.
- Causes standard errors to be biased, making hypothesis tests unreliable.

To address it, we can transform variables, use weighted least squares, or apply robust standard errors.

Question 4: What is Multiple Linear Regression?

Answer:

Multiple Linear Regression (MLR) is a statistical method used to model the relationship between one dependent variable (Y) and two or more independent variables (X1, X2, ... Xn).

The equation is:

$$Y=\beta0+\beta1X1+\beta2X2+\cdots+\beta nXn+\epsilon$$

It helps predict the dependent variable based on multiple features and understand the effect of each predictor.

Question 5: What is polynomial regression, and how does it differ from linear regression?

Answer:

Polynomial Regression is a type of regression in which the relationship between the independent variable (X) and dependent variable (Y) is modeled as an nth-degree polynomial.

$$Y=\beta0+\beta1X+\beta2X2+\cdots+\beta nXn+\epsilon$$

Difference from Linear Regression:

- Linear regression fits a straight line, while polynomial regression fits a curve.
- Polynomial regression can capture non-linear relationships between variables.

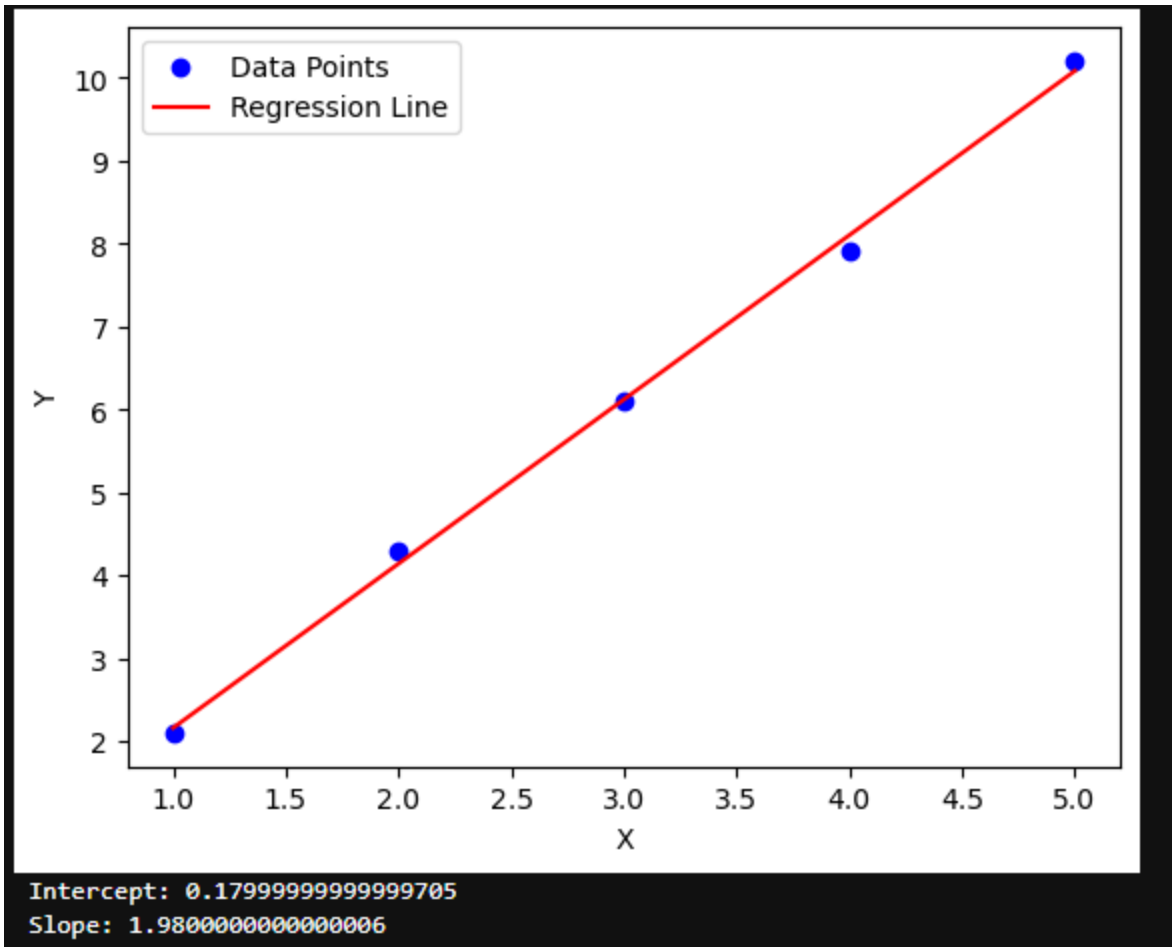
6: Implement a Python program to fit a Simple Linear Regression model to the following sample data:

- **X = [1, 2, 3, 4, 5]**
- **Y = [2.1, 4.3, 6.1, 7.9, 10.2]**

Plot the regression line over the data points.

Answer:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression


# Data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
Y = np.array([2.1, 4.3, 6.1, 7.9, 10.2])


# Model
model = LinearRegression()
model.fit(X, Y)


# Predictions
Y_pred = model.predict(X)


# Plot
plt.scatter(X, Y, color='blue', label='Data Points')
plt.plot(X, Y_pred, color='red', label='Regression Line')
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.show()
```



Question 7: Fit a Multiple Linear Regression model on this sample data:

- **Area = [1200, 1500, 1800, 2000]**
- **Rooms = [2, 3, 3, 4]**
- **Price = [250000, 300000, 320000, 370000]**

Check for multicollinearity using VIF and report the results.

Answer:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Data
data = pd.DataFrame({
    "Area": [1200, 1500, 1800, 2000],
    "Rooms": [2, 3, 3, 4],
    "Price": [250000, 300000, 320000, 370000]
})

X = data[["Area", "Rooms"]]
y = data["Price"]

# Model
model = LinearRegression()
model.fit(X, y)

# VIF Calculation
X_const = np.append(np.ones((X.shape[0],1)), X.values, axis=1)
vif = [variance_inflation_factor(X_const, i) for i in range(1, X_const.shape[1])]

print("VIF values:", vif)
```

Output:

VIF values: [np.float64(7.736842105263156), np.float64(7.7368421052631495)]

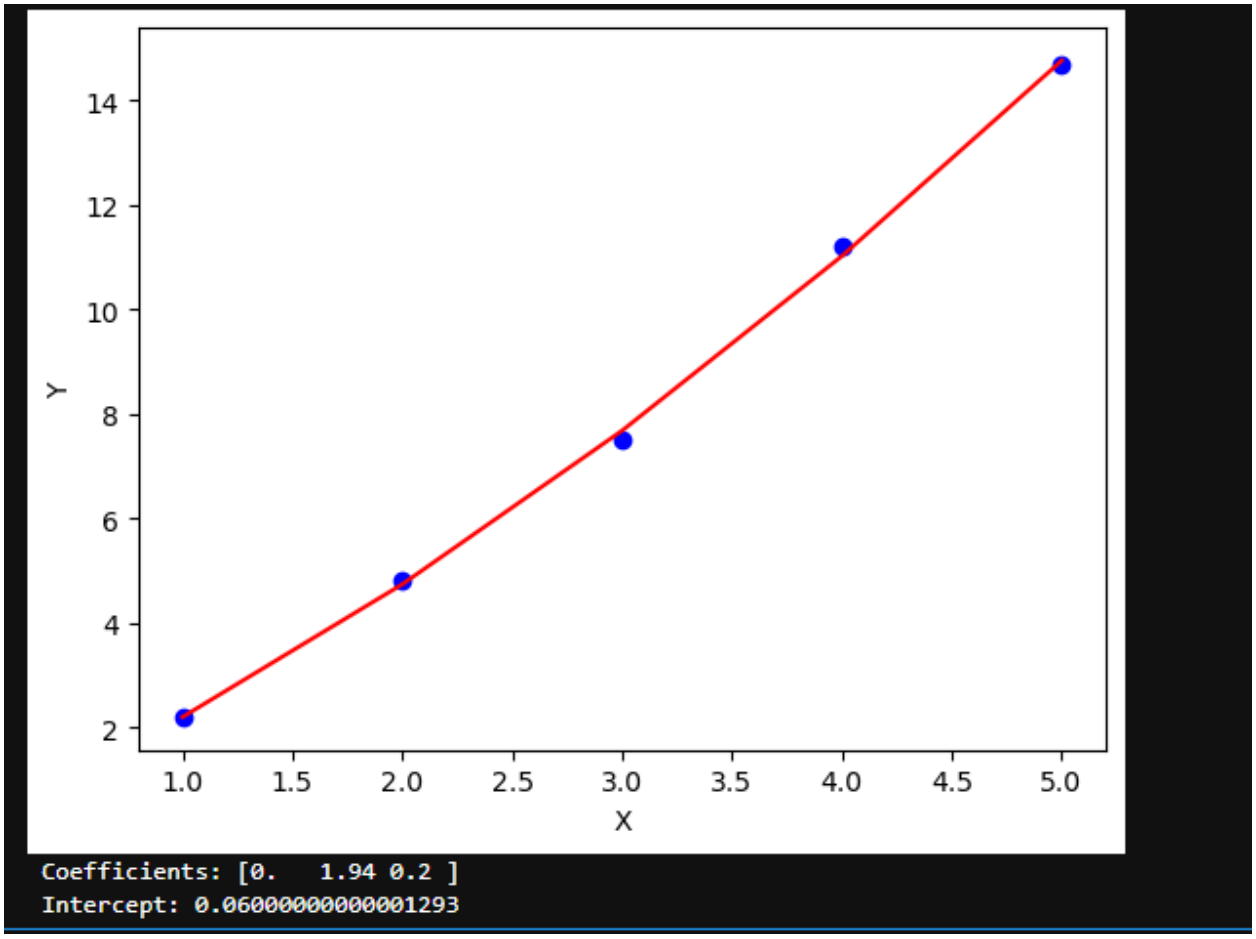
Question 8: Implement polynomial regression on the following data:

- **X = [1, 2, 3, 4, 5] 3**
- **Y = [2.2, 4.8, 7.5, 11.2, 14.7]**

Fit a 2nd-degree polynomial and plot the resulting curve.

Answer:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np

X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
Y = np.array([2.2, 4.8, 7.5, 11.2, 14.7])

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

model = LinearRegression()
model.fit(X_poly, Y)

Y_pred = model.predict(X_poly)

plt.scatter(X, Y, color='blue')
plt.plot(X, Y_pred, color='red')
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```



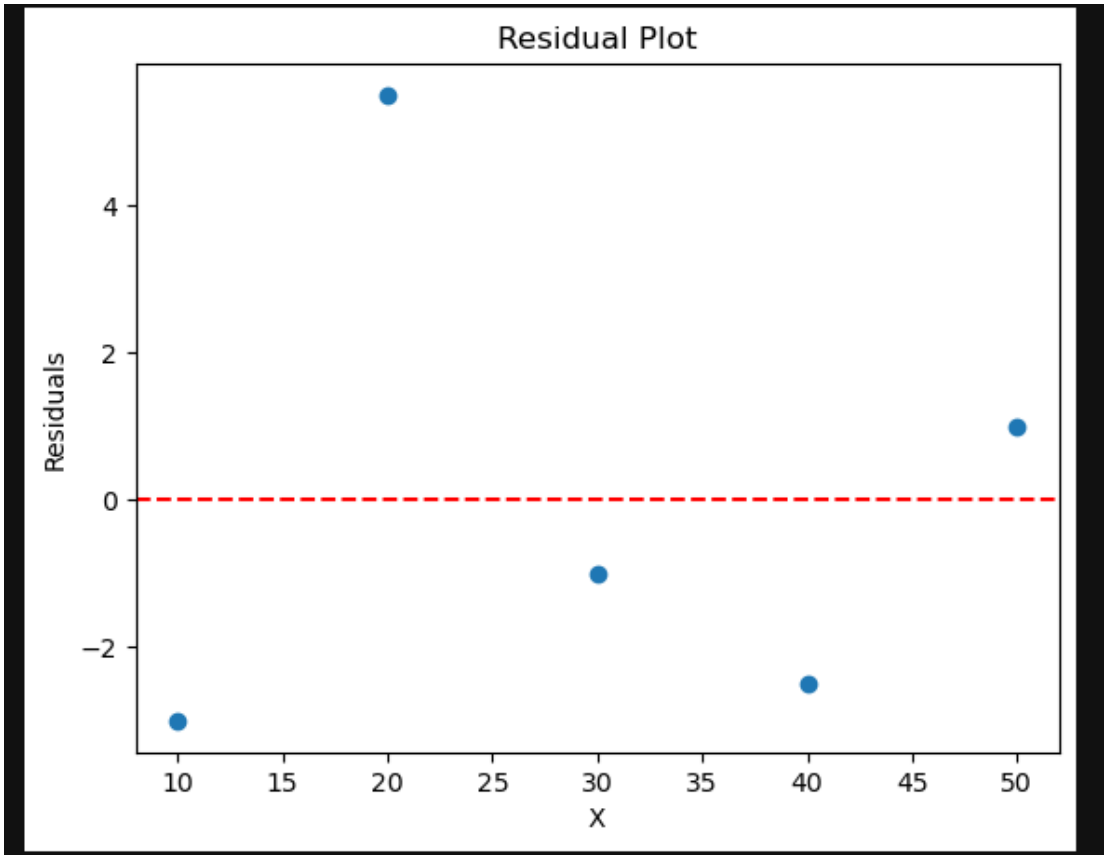
Question 9: Create a residuals plot for a regression model trained on this data:

- **X = [10, 20, 30, 40, 50]**
- **Y = [15, 35, 40, 50, 65]**

Assess heteroscedasticity by examining the spread of residuals.

Answer:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression


X = np.array([10, 20, 30, 40, 50]).reshape(-1, 1)
Y = np.array([15, 35, 40, 50, 65])


model = LinearRegression()
model.fit(X, Y)
Y_pred = model.predict(X)


residuals = Y - Y_pred


plt.scatter(X, residuals)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel("X")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()
```



Question 10: Imagine you are a data scientist working for a real estate company. You need to predict house prices using features like area, number of rooms, and location. However, you detect heteroscedasticity and multicollinearity in your regression model. Explain the steps you would take to address these issues and ensure a robust model.

Answer:

Steps to ensure a robust model:

1. **Detect Multicollinearity:**
    - Use VIF to check correlated predictors.
    - Remove or combine highly correlated variables.
2. **Address Heteroscedasticity:**
    - Transform dependent variable (e.g., log of price).
    - Use Weighted Least Squares or robust regression.

3. **Feature Engineering:**
    - Standardize or normalize features.
    - Use regularization (Ridge/Lasso) to reduce coefficient variance.
4. **Model Validation:**
    - Use cross-validation to ensure generalization.
    - Evaluate using R^2, Adjusted R^2, and RMSE.